REMARKS

This is filed in connection with a Request for Continued Examination filed this day herewith, and in response to the final Office Action dated April 1, 2009, rejecting claims 1 – 63 under 35 U.S.C. §103. In view the remarks that follow, and the amendments above, the Applicants submit that all pending claims are in condition for allowance.

I. The Pending Claims are Patentably Distinct from the Art

A. Claims 1 – 10, 13 – 19, 23 – 43, 46 – 49 and 52 – 63

Claims 1 – 63 stand rejected under 35 U.S.C. § 103(a) as being allegedly unpatentable over Emer et al. (US 6,493,741; hereinafter "Emer") in view of Kelsey et al. (US 7,082,519; hereinafter "Kelsey"), which incorporates by reference Eggers (Eggers et al. "Simultaneous Multithreading: A Platform for Next-Generation Processors, IEEE, 1997; pages 12 – 19), and Sekiguchi et al. (US 2001/0016879; hereinafter "Sekiguchi").

Claim 1 is directed to an embedded processor comprising a plurality of processing units that each execute processes or threads (collectively, "threads"). One or more execution units are shared by the processing units and execute instructions from the threads. An event delivery mechanism is in communications coupling with the plurality of processing units and delivers to respective threads, without execution of instructions by the processing units, hardware interrupts from other devices and/or software interrupts from other threads.

Neither Emer, Kelsey nor Sekiguchi, individually and in combination, teach or suggest an embedded processor meeting the limitations of claim 1.

For example, Emer purports to disclose a method for resolving the problem of spin-lock in an SMT architecture by halting, or "quiescing," threads while they are waiting for a lock. *See* Emer, col. 3, lines 24 – 27. Nowhere does Emer discuss interrupt delivery, much less, an event delivery mechanism that delivers to respective threads hardware interrupts from other devices and/or software interrupts from other threads, e.g., as required by amended claim 1. Nor,

consequently, does Emer suggest how interrupts might be delivered without execution of instruction by the processing units.

To the contrary, Emer's disclosure is limited to use of an "event monitor" within each so-called thread processing unit to compare an address on a memory bus with an address stored in a local memory, as shown in Figure 7 of that patent and discussed in the corresponding text, both reprinted below:

> FIG. 7 is a schematic diagram illustrating an embodiment of the present invention in which the event monitor 109 watches memory address lines 135A and control lines 135B. Comparator 180 compares the watch_physical_address 103 with the address on the memory bus 135 address lines 135A. The comparator 180 is only enabled for write operations, for example, when WRITE is asserted on a read/write control line 135B. The output 181 of the comparator 180 indicates whether a write to the identified location, i.e., the monitored event, has occurred.

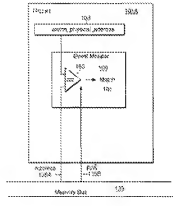


Fig. 7

Emer, at col. 8, lines 27, et seq,

Use of the match signal generated by the comparator of Emer's event monitor is limited to the very thread processing unit and for the very thread for which the comparison is made. This is evident in Emer's Figure 6, which shows a separate event monitor for each thread processing unit:

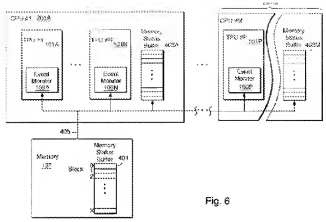


Fig. 6

It follows that Emer is devoid of suggestion that the match signal (or any other output) of an event monitor could be used outside the very thread processing unit or the very thread for which it was generated. Any such discussion in regard to interrupts is, of course, wholly lacking as well.

Kelsey and Sekiguchi do not remedy the deficiencies of Emer. For example, the Office Action suggests that Kelsey teaches an event delivery mechanism capable of delivering events to threads with which they are associated. *See*, Office Action, mailed April 1, 2009, pages 4 – 5. However, the cited passages of Kelsey, col. 4, lines 11 – 19, and col. 8, lines 60 – 67, reprinted below, teach nothing of the sort:

The invention is a system and method for the enabling multithreading in a embedded processor, invoking zero-time context switching in a multithreading environment, scheduling multiple hardware threads to permit numerous hardreal time and non-real time priority levels, fetching data and instructions from multiple memory blocks in a multithreading environment, and enabling a particular thread to store multiple states of the multiple threads in the instruction pipeline.	The present invention includes hardware support for running multiple software threads and automatically switching between threads and is described below. This multithreading support includes a variety of features including real time and non-real time task scheduling, inter-task communication with binary and counting semaphores (interrupts), fast interrupt response and context switching, and incremental linking.
Kelsey at col. 4, lines 11 – 19.	Kelsey at col. 8, lines 60 – 67.

As readily evident on review of the quoted text, there is no mention of event delivery, much less interrupt delivery. Indeed, it appears that the words "deliver," "delivery," or the like, are not used in the patent at all. Nor does there appear to be any discussion of the concept of interrupt delivery.

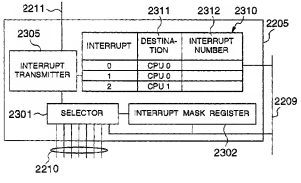
Not only does Kelsey fail to teach event delivery, that publication also fails to teach or suggest, *inter alia,* (i) an event delivery mechanism that delivers to respective threads hardware interrupts from other devices and/or software interrupts from other threads, e.g., as required by amended claim 1, nor (ii) delivery of interrupts without execution of instructions by the processing units.

Sekiguchi does not remedy these deficiencies. That publication purports to disclose a method and system for configuring a plurality of operating systems. *See* Sekiguchi, ¶ 0015. However, it too fails to teach or suggest, *inter alia,* an event delivery mechanism capable of delivering events to threads without execution of instructions by the processing units, e.g., as required by pending claim 1.

The Office Action incorrectly asserts that Sekiguchi "teaches an event delivery mechanism that delivers events to threads with which those events are associated, wherein the event delivery mechanism delivers each such event to the respective thread without execution of instructions by said processing units." *See* Office Action, mailed April 1, 2009, page 5. However, the Office Action unfairly attributes to the Sekiguchi interrupt controller capabilities that simply are not disclosed in that publication.

Specifically, for example, nowhere does Sekiguchi teach or suggest that the interrupt controller can deliver events to threads. At most, that publication suggests that the controller can specify a CPU destination for an interrupt — not a particular thread, process or otherwise! For example, Figure 23 of Sekiguchi, reprinted left, clearly shows that the only destination specified for an interrupt is a CPU. There is no indication of any capability for delivery to a thread, process, or otherwise.

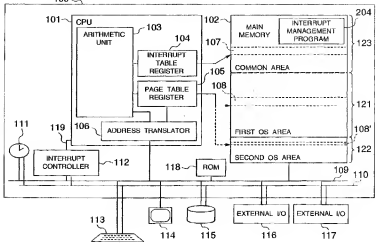
FIG. 23



Indeed, closer scrutiny of Sekiguchi reveals that processor instructions are actually required for interrupt delivery. Referring to Figure 1 of that publication, reprinted below, there is shown, among other things, an interrupt table register 104 within the CPU 101. In regard to that drawing, Sekiguchi says "[w]hen an interrupt occurs, the processor 101 receives the interrupt

number from the interrupt controller 112. By using this number as a search index, the processor acquires an interrupt handler address from the interrupt table 107 to pass the control to the interrupt handler." *See* Sekiguchi, ¶ 0050, lines 10 – 15.

FIG. 1



It is thus quite clear that Sekiguchi requires processor instructions to deliver interrupts. This is contrary to the recitation of claim 1, which calls for delivery of events to respective threads without execution of instructions by the processing units.

Not only do Emer, Kelsey and Sekiguchi fail to individually teach or suggest the subject matter of claim 1, their combined teachings are equally lacking. Thus, for example, were a person of ordinary skill in the art to come upon the notion of using Sekiguchi's interrupt delivery mechanism to distribute notifications generated by Emer's event monitor comparator, the resultant device would fall far short of the claimed invention. By way of non-limiting example, that resultant device would still require software (as does Sekiguichi) to deliver those notifications.[1] Likewise, were a person of ordinary skill in the art combine to come upon the notion of using Emer's event monitor comparator to "compare" interrupts from Sekiguchi's interrupt controller, the resultant device would *inter alia* lack an ability to deliver interrupts to any of a plurality of processing units, each of which execute one or more threads and one or more of which execute a plurality of threads. Instead, such a device — if it were operative at all! — could hardly be characterized as distributing interrupts in the conventional sense and, moreover, would require software to discern to which of multiple threads to deliver an interrupt.

[1] The Applicant's do not concede that persons of ordinary skill in the art would be even be led to make such a combination, shy of impermissibly using Applicant's claimed invention as a roadmap.

In view of the foregoing, it is evident that the combination of Emer, Kelsey and Sekiguchi fails to teach, suggest or otherwise render unpatentable the subject matter of claim 1. The same is true for claims 2 – 6 which depend from claim 1 and recite further limitations thereon.

For like reasons, Emer, Kelsey and Sekiguchi fail to teach, suggest or otherwise render unpatentable the subject matter of claims 7, 17, 28, 32, 35, 41, 47, 57 and 61, which parallel claim 1 in regards relevant to the arguments above. The remaining claims depend from either claim 7, 17, 28, 32, 35, 41, 47, 57 or 61, reciting further limitations thereon. For at least the reasons above, they too are patentable over Emer, Kelsey and Sekiguchi.

B. Claims 11, 12, 20 – 22, 44, 45, 50 and 51

Claims 11, 12, 20 – 22, 44, 45, 50 and 51 stand rejected under 35 U.S.C. § 103(a) as being allegedly unpatentable over Emer in view of Kelsey, which incorporates Eggers, and Sekiguchi, and further in view of Microsoft Computer Dictionary (Fifth Edition, Microsoft Press, 2002). These claims depend from claims 7, 17, 41 and 47, and are patentably distinct from the teachings of Emer in view of Kelsey and Sekiguchi for at least the reasons discussed above in connection with claim 1. Insofar as Microsoft Computer Dictionary fails to remedy the deficiencies of those other references — namely, failing to teach or suggest *inter alia* an event delivery mechanism that is in communication coupling with a plurality of (virtual) processing units and that delivers events to respective threads with which those events are associated without execution of instruction by said processing units — claims 11, 12, 20 – 22, 44, 45, 50 and 51 are patentably distinct from the combination of Emer, Kelsey and Microsoft Computer Dictionary.

C. Claims 26 and 55

Claims 26 and 55 stand rejected under 35 U.S.C. § 103(a) as being allegedly unpatentable over Emer in view of Kelsey, which incorporates Eggers, and Sekiguchi, and further in view of Blandy (US 6,912,647). These claims depend from claims 17 and 47, respectively, and are

patentably distinct from the teachings of Emer in view of Kelsey and Sekiguchi for at least the reasons above. Insofar as Blandy fails to remedy the deficiencies of those other references — namely, failing to teach or suggest *inter alia* an event delivery mechanism that is in communication coupling with a plurality of (virtual) processing units and that delivers events to respective threads with which those events are associated without execution of instruction by said processing units — claims 26 and 55 are patentably distinct from the combination of Emer, Kelsey and Blandy.

II. New Claims 67 and 68

New claims 67 and 68 parallel claims 28 and 57, albeit reciting that the event delivery mechanism delivers to respective threads (without execution of instructions by the processing units) events that comprise (i) loading of cache memory following a cache miss by the thread to which that event is delivered, (ii) filling of a memory location by a thread, other than the thread to which that notification is delivered, in response to a memory instruction issued by the thread to which that notification is delivered, and wherein the event delivery mechanism. For reasons paralleling those discussed above, these claims, too, are patentably distinct from the art.

II. Conclusion

In view of the foregoing, the Applicants believe that the application is in condition for allowance. The Examiner is encouraged to telephone the undersigned attorney for Applicants if such communication will expedite prosecution of this application.

Respectfully submitted,

Date: August 7, 2009

/Benjamin E. Berman/

Benjamin E. Berman (Reg. 61,507)
David J. Powsner (Reg. No., 31,868)
Attorney for Applicants
Nutter McClennen & Fish LLP
World Trade Center West
155 Seaport Boulevard
Boston, MA 02210-2604
Tel: (617) 439-2000
Fax: (617) 310-9000